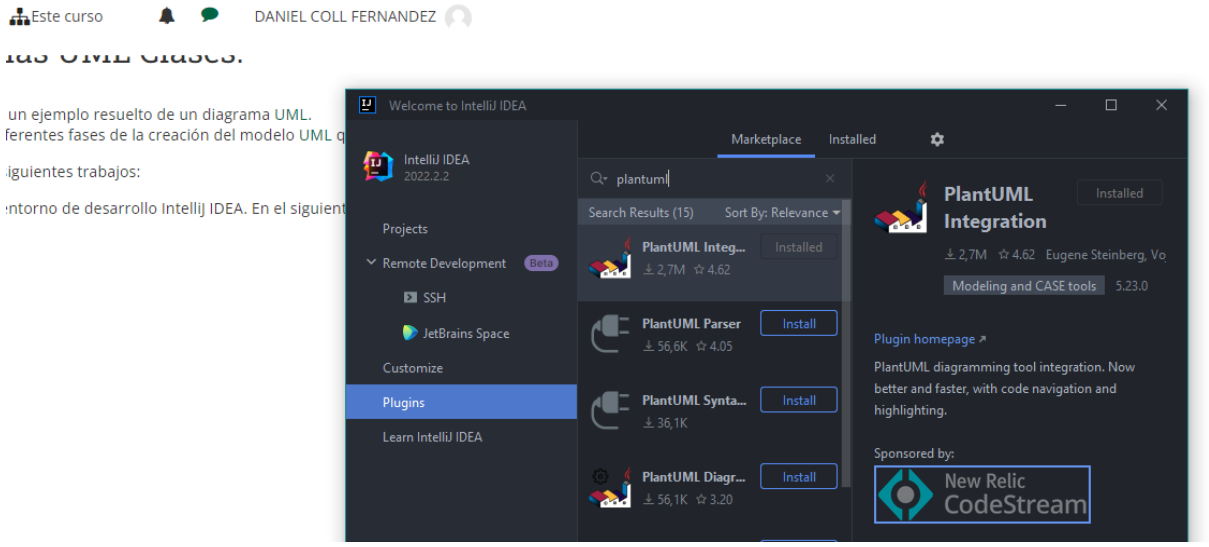
Entornos de desarrollo

Práctica diagrama de clases UML

En esta práctica, voy a seguir el ejemplo resuelto propuesto en aules para completar un ejercicio de desarrollo de diagramas de clase utilizando el plugin PlantUML integrado en el entorno IntellIJ.

El primer paso para esto es instalar dicho plugin, así que desde la pestaña de inicio de Intellij me dirijo al apartado de plugins, busco plantuml y lo instalo:



He creado un repositorio en github llamado UMLClasesDanielColl en el que iré guardando el proceso y donde se podrá apreciar el resultado final. Este es el enlace: https://github.com/Daniclo/UMLClasesDanielColl

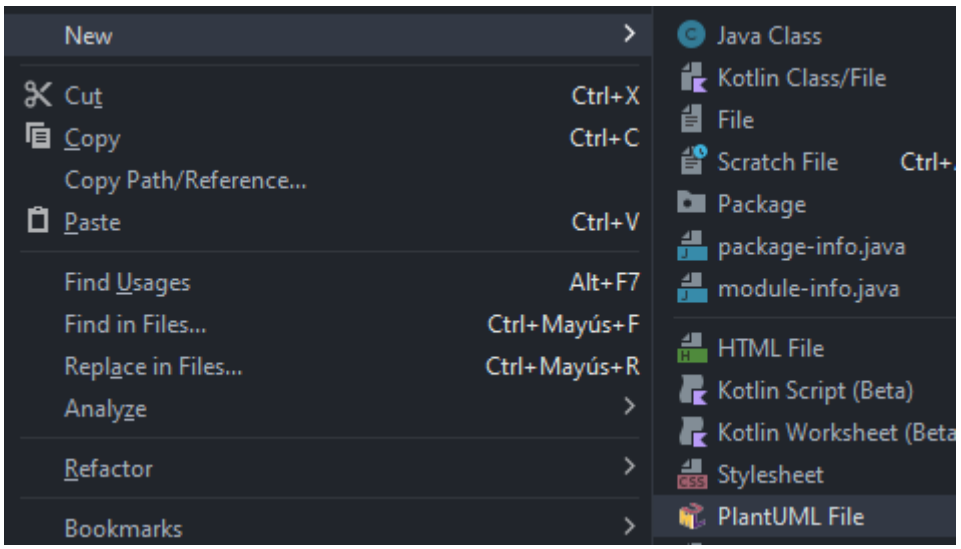
Ahora si voy a empezar con la práctica. El enunciado nos proporciona información sobre un diagrama de clases que tenemos que generar para realizar el programa que nos ha pedido una asociación para ayudarla a gestionar sus datos:

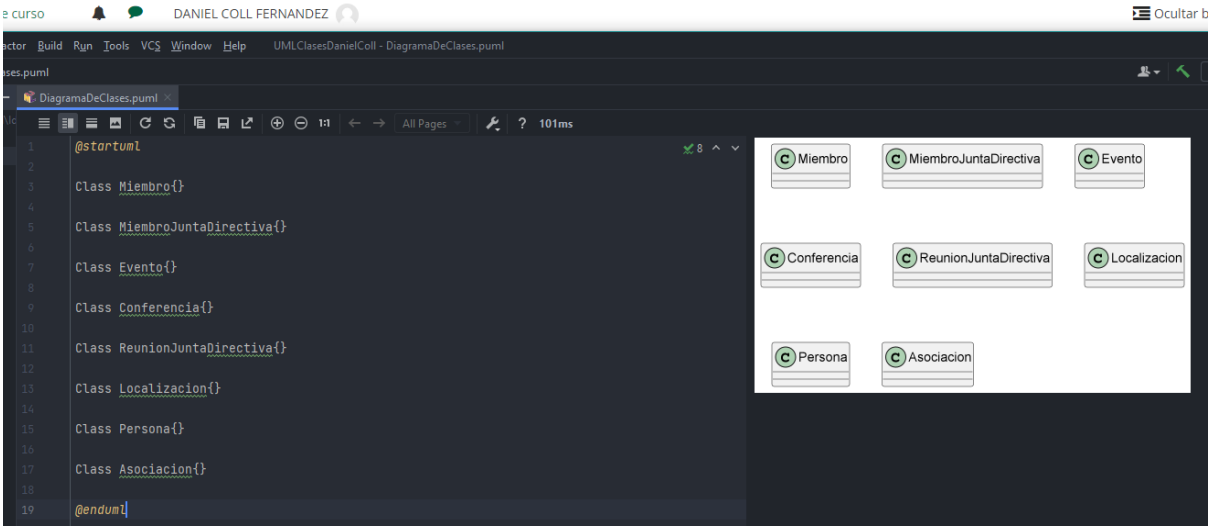
> La Asociación de Antiguos Alumnos de la UOC nos ha pedido si podemos ayudarles a confeccionar un programa que les permita gestionar a sus asociados, eventos y demás elementos relacionados.
>
> Los asociados se pueden dividir en miembros numerarios y en miembros de la junta directiva, que es elegida por votación en una asamblea general cada cuatro años. La única diferencia entre ellos es que los miembros de la junta directiva son convocados a las reuniones de junta y los demás miembros no, pero el resto de actividades que se realizan están abiertas a todos los miembros de la asociación.
>
> La convocatoria de un evento se realiza por correo electrónico a todos los miembros activos en el momento del envío, recibiendo un enlace para aceptar su participación. En todos los eventos, la aceptación de los asistentes se realiza por orden de llegada ya que, en algún caso, se puede dar que el número de asistentes sea limitado, como en las conferencias.
>
> En la convocatoria, también aparece información sobre el lugar que en muchos casos se repite, por lo que nos han dicho que quieren almacenar los datos para futuros usos.

En base a este enunciado, el primer paso es identificar las clases que van a componer mi diagrama. En este caso serán las siguientes:

- Miembro
- Miembro de la junta directiva
- Evento
- Conferencia
- Reunión de la junta directiva
- Localización
- Persona
- Asociación (AAUOC)

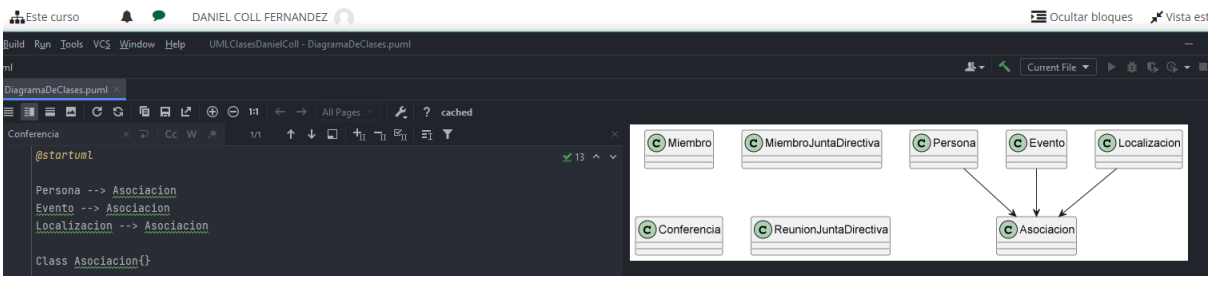
Una vez identificadas las clases que hacen falta, voy a crearlas en plantUML de la siguiente forma. Primero creo un archivo de PlantUML. Aparecerá un código de ejemplo pero voy a eliminarlo para escribir desde 0. Para crear una clase simplemente escribo class, el nombre de la clase y añado unas llaves:



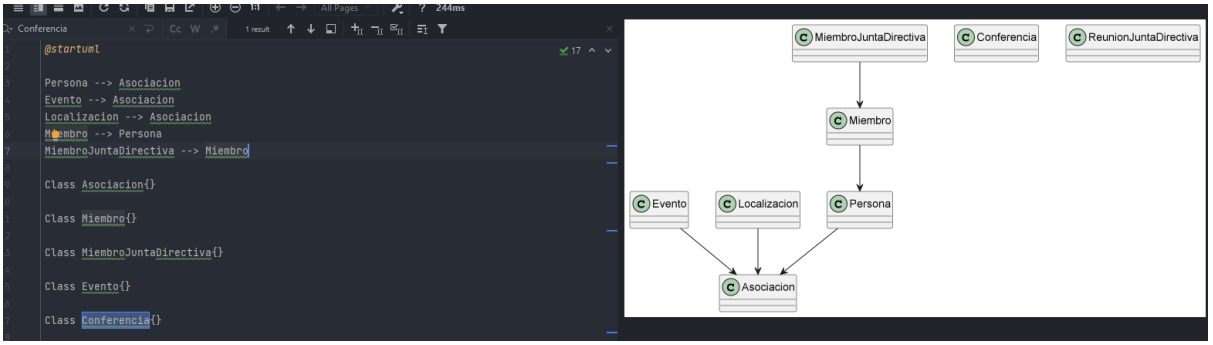


Pero estas clases están vacías y no interaccionan entre ellas. Eso me lleva directamente al siguiente paso. Toca decidir las interacciones y relaciones entre estas clases que he creado.

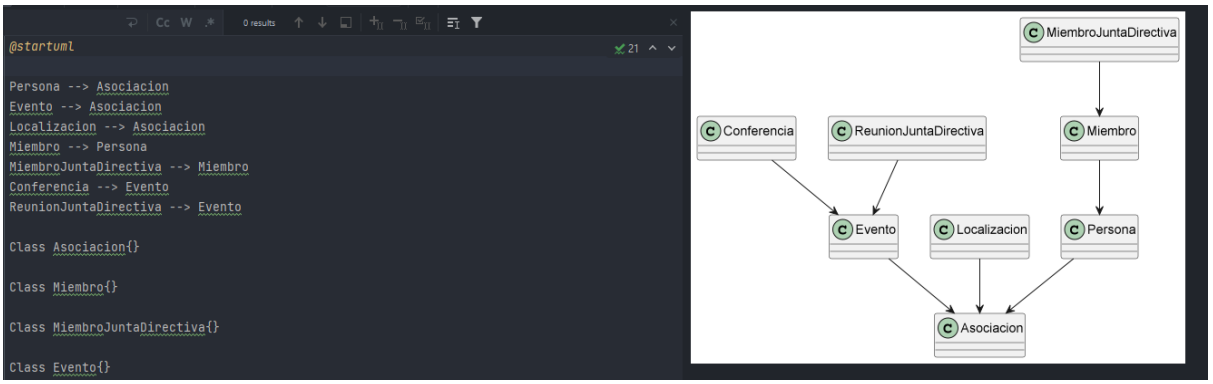
La clase Asociación va a ser la clase principal del proyecto y de ella van a heredar las clases Persona, Localización y Evento:



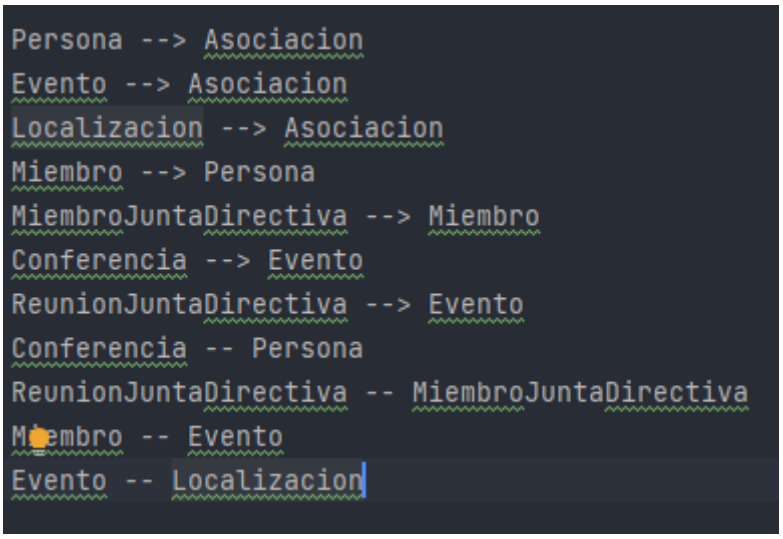
Desde la clase Persona nacerá la subclase Miembro y como subclase de esta, MiembroJuntaDirectiva:

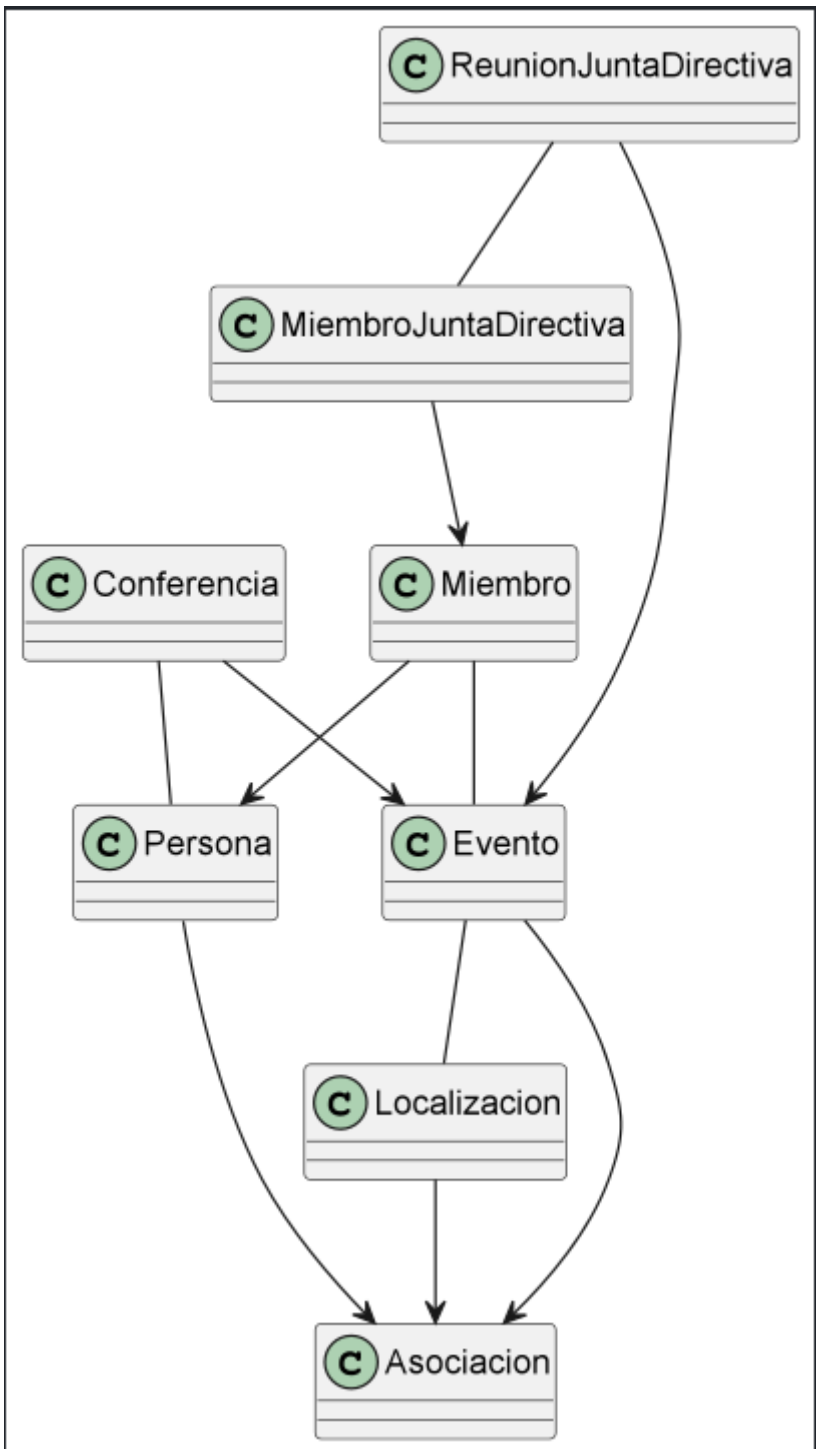


De la clase Evento nacen 2 subclases: Conferencia y ReuniónJuntaDirectiva:



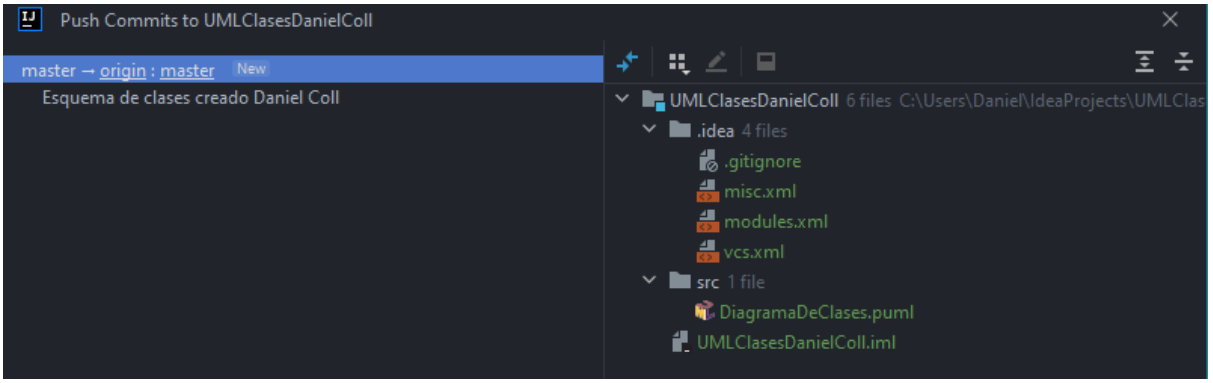
Y por último añadimos algunas relaciones: Conferencia con Persona (personas que atienden/dan la conferencia), ReunionJuntaDirectiva con MiembroJuntaDirectiva (miembros de la junta atienden a la reunión/la convocan), Miembro con Evento (los miembros de cualquier tipo atienden a eventos de cualquier tipo) y Evento con Localización (cada evento tiene un lugar). Estas relaciones las completaré más adelante:



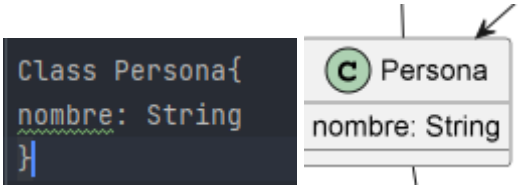


Este es el resultado de las interacciones del diagrama.

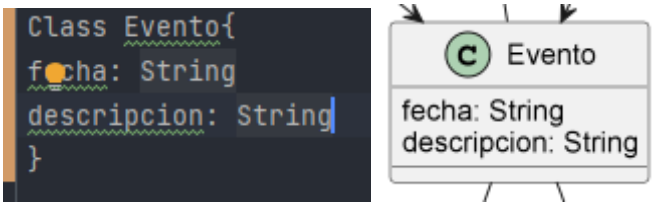
Ahora, para continuar, voy a completar cada clase con sus atributos y métodos. No sin antes hacer el primer commit del proyecto:



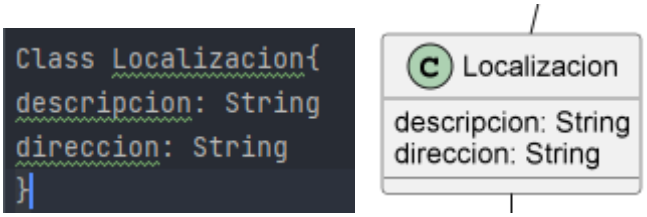
Voy a comenzar con los atributos. De cada Persona queremos almacenar su nombre:



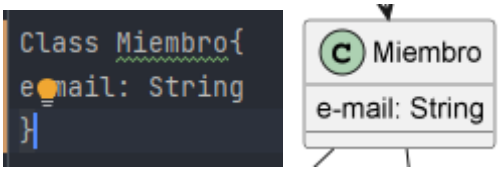
De cada evento vamos a incluir los datos fecha y descripción:



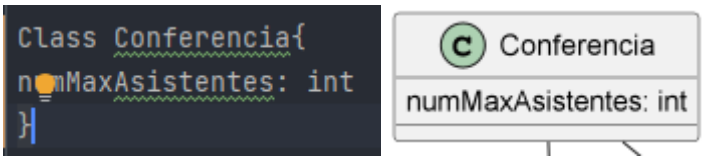
Para las localizaciones quiero almacenar una descripción y una dirección:



De los miembros nos interesa el nombre, pero no hace falta añadirlo porque heredan de Persona, y adicionalmente, el email:



Y por último, en las conferencias vamos a almacenar, además de los atributos propios de los eventos, el número máximo de asistentes:

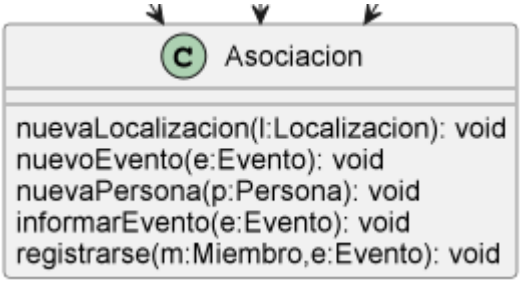


Estos serían todos los atributos que este programa va a almacenar. Para los métodos, la asociación de antiguos alumnos nos ha dejado estas indicaciones:

> La asociación necesitará un conjunto de métodos para añadir nuevos eventos, personas y localizaciones al sistema (métodos *newX* de la clase *AAUOC*), así como también un método para informar a los miembros de la convocatoria de un evento (método *informEvent*). Al mismo tiempo, se dice que los usuarios necesitarán confirmar la asistencia a los eventos (método *register*, que deberá almacenar los asistentes por orden y controlar el número máximo de éstos si fuera necesario).

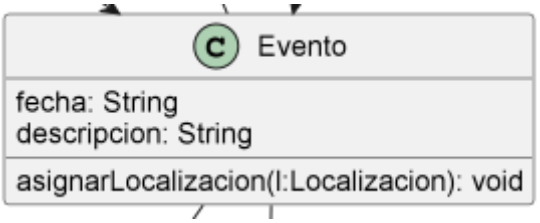
Voy a comenzar por la clase Asociación, desde la cual se van a poder añadir Personas, Localizaciones y Eventos nuevos con 3 métodos diferentes. También se podrá informar a los miembros de que existe un nuevo evento con otro método y un último permitirá a estos confirmar su asistencia:

```
Class Asociacion{
nuevaLocalizacion(l:Localizacion): void
nuevoEvento(e:Evento): void
nuevaPersona(p:Persona): void
informarEvento(e:Evento): void
registrarse(m:Miembro,e:Evento): void
}
```

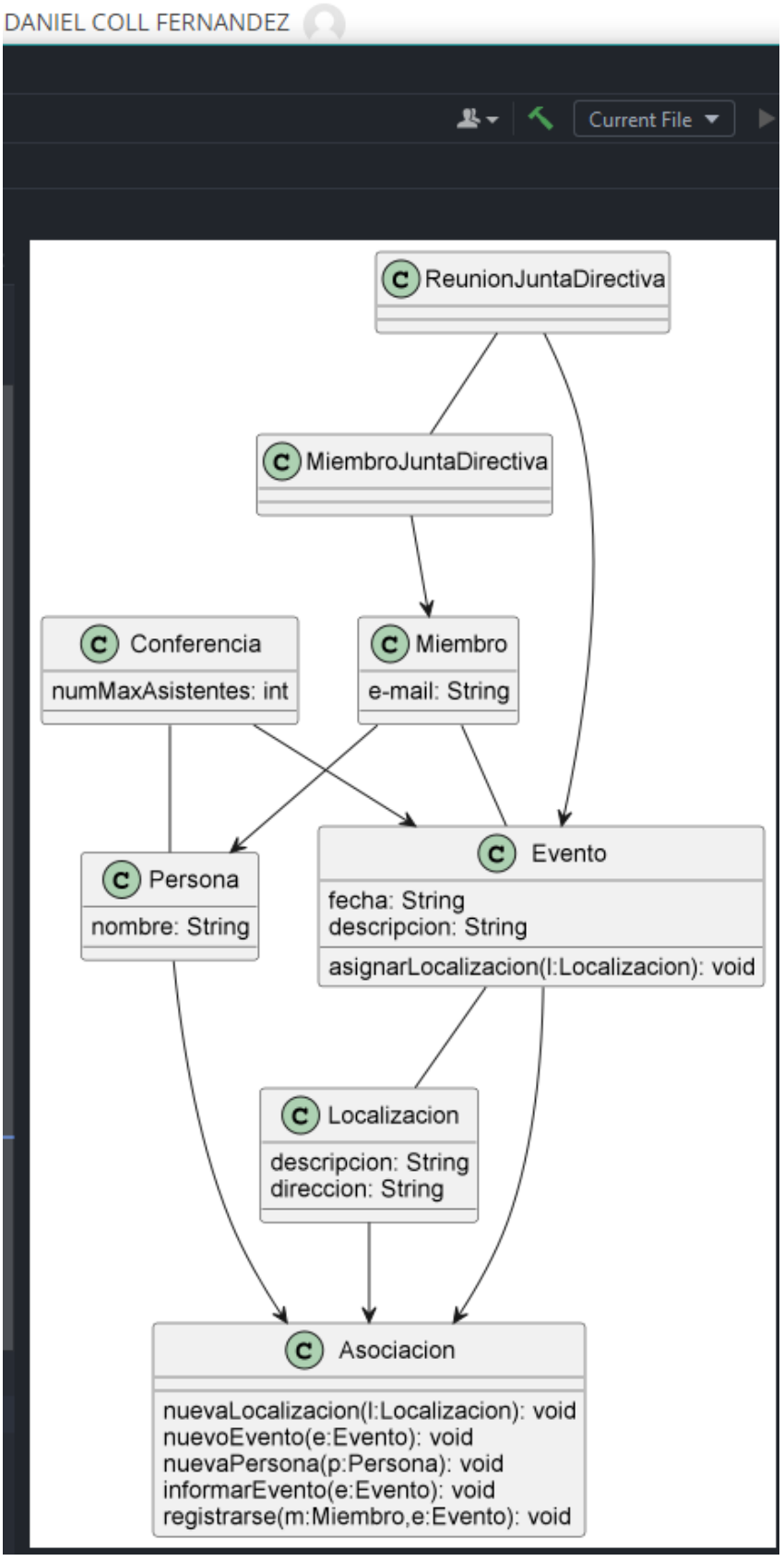


También voy a añadirle a la clase Evento un método que permita asignar a cada evento una localización:

```
Class Evento{
fecha: String
descripcion: String
asignarLocalizacion(l:Localizacion): void
}
```



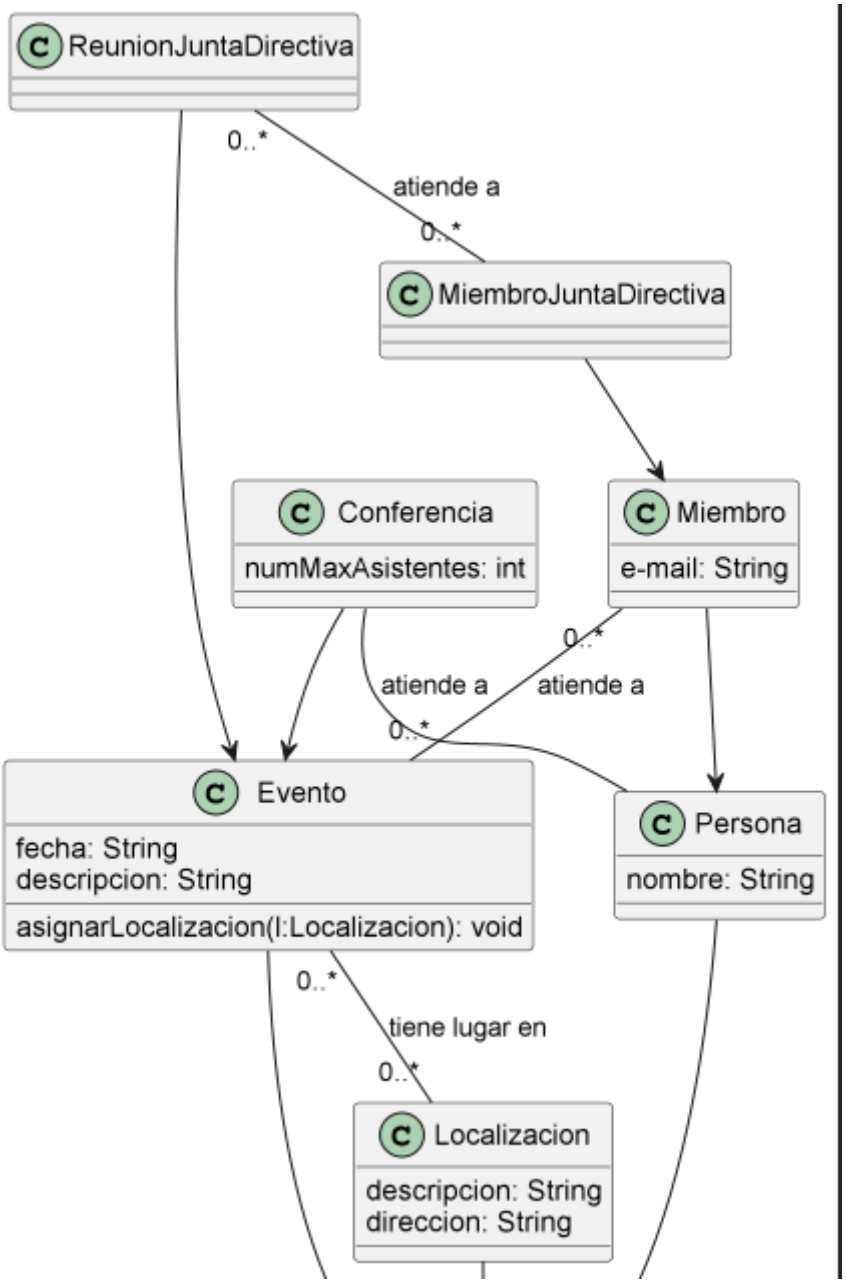
Con esto ya están añadidos todos los atributos y métodos de cada clase, y por ahora el diagrama se ve así:



El último paso para completar este diagrama es establecer la cardinalidad y navegabilidad de las relaciones. Como no hay indicaciones al respecto, todas las relaciones serán bidireccionales.

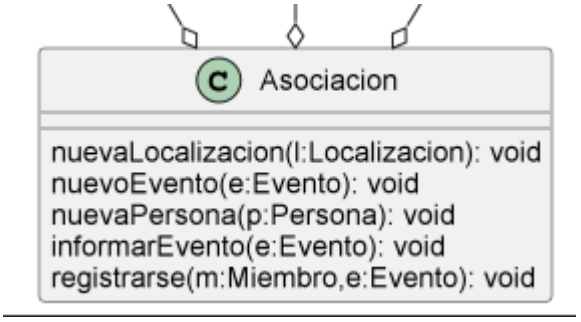
Comenzaré por completar las relaciones que creé antes que no eran herencias:

```
Conferencia -- Persona : atiende a
ReunionJuntaDirectiva "0..*" -- "0..*" MiembroJuntaDirectiva : atiende a
Miembro "0..*" -- "0..*" Evento : atiende a
Evento "0..*" -- "0..*" Localizacion : tiene lugar en
```

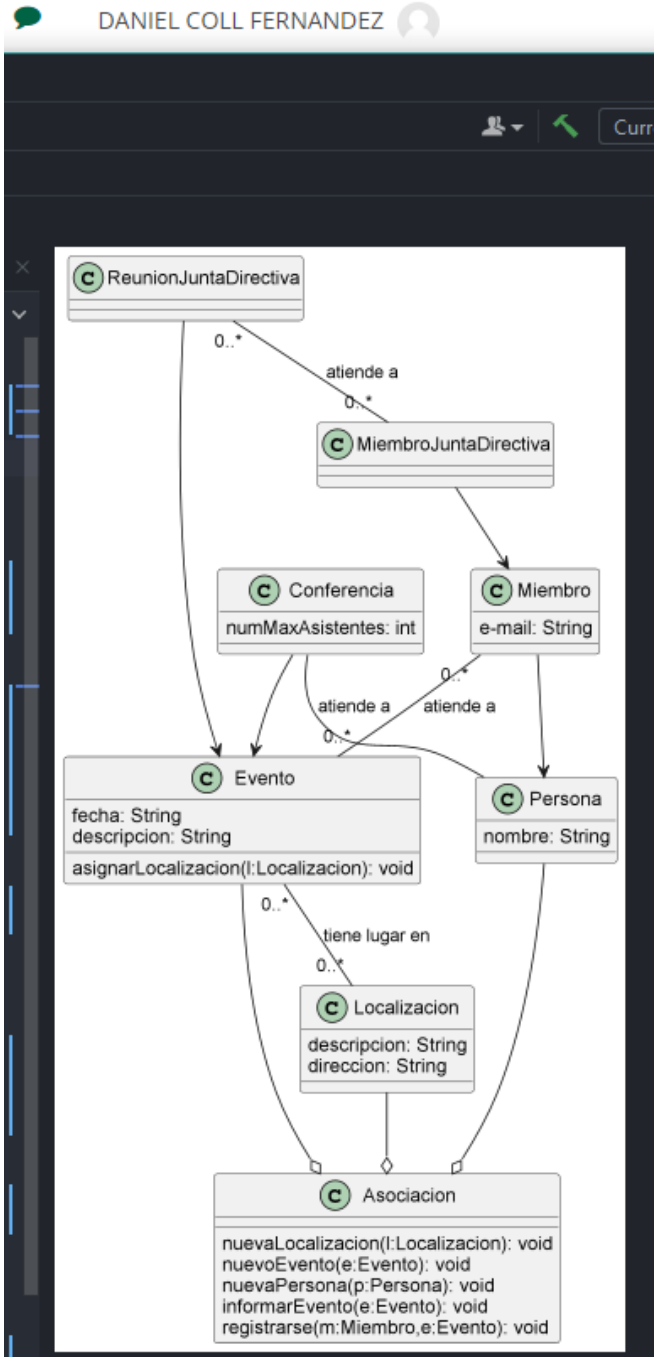


Ahora voy a sustituir las flechas que apuntan desde Evento, Persona y Localización hasta Asociacion por diamantes de relación, como indica en la tarea guiada:

```
Persona --o Asociacion
Evento --o Asociacion
Localizacion --o Asociacion
```



Y con esto, el diagrama de clases ya cumple todos los requisitos que exige la asociación. El resultado final del diagrama es el siguiente:



Para finalizar, realizo el último commit y el último push a github, y adjunto este documento en formato doc y en formato pdf al repositorio remoto: